Augmented Reality Indoor Positioning and Navigation

Anna Horwath, Ernests Lavriovics, Christian Neurohr

Abstract

Indoor navigation is a topic undergoing research within robotics, but can also be applied in commerce, manufacturing or industrial applications. We investigate how AR Head-Mounted Display Magic Leap 1 can be integrated into an indoor navigation framework aimed for commerce or asset management. Magic Leap 1 supports spatial mapping of real-life environment as well as tracking users in six degrees-of-freedom, and provides the ability of recognizing and estimating the orientation in space of fiducial markers. We propose an end-to-end framework that is able of mapping out an environment, digitally processing it and providing a framework for connecting client devices for issuing navigation requests. Our system proves to function end-to-end although fails to improve task completion time when tested in a mock-up environment. This can be attributed to over-simplified test task, low resolution of guidance elements and general inexperience with AR devices by the test subjects. Our study concludes that the first iteration design should have an improved user interaction and it should be tested with a different methodology. It also identifies future work and features such as IoT communication, computer vision integration and a revamped user interface based around human-computer interaction research.

1 Introduction

Modern mainstream navigation services such as Google Maps, Apple Maps, Waze are all excellent ways of getting around outdoor environments down to the precision of 5 meters under open sky [1]. Additionally, Google Maps also has already deployed in production an Augmented Reality (AR) solution called *Live View* [2] that is able to annotate directions on a pass-through video using a modern smartphone, and a beta feature is available on certain devices that allows for locating people in a crowd upon subject's permission.

Tech companies such as Google and Apple have also started initiatives for indoor maps [3] [2] although they still use GPS as the main navigation engine and the frameworks do not scale to more specific contexts such as navigating to particular items within warehouses or shops. Previous works done within the field of indoor positioning agree that GPS is not enough for localizing in an indoor environment [4] [5]. Other approaches besides GPS for localizing a user would be communication-based such as Bluetooth, Ultra-Wide Band(UWB), Visible-Light Communication (VLC) or Wi-Fi [5] although in some cases these can be expensive, may require specialized equipment and create an overall burden by extending the number of hardware units necessary within a given system, overview of accuracy vs cost is depicted in Figure 1.

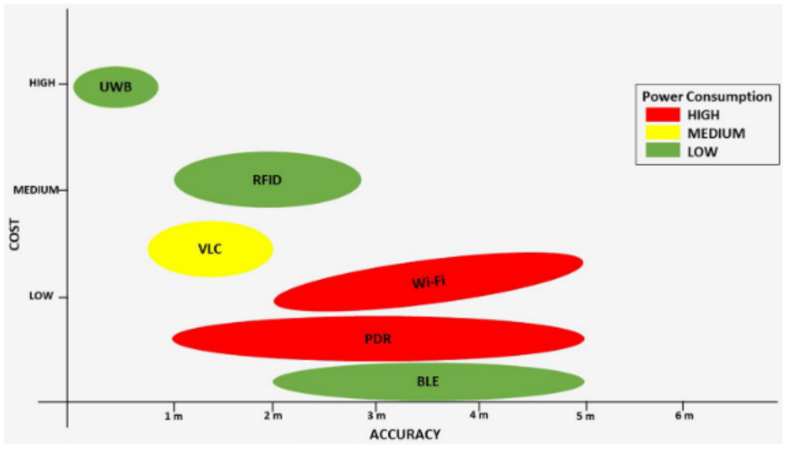


Figure 1: Overview of Communications Based Tracking [5]

A technique for reducing the hardware units necessary for a navigational system would be to incorporate Simultaneous Mapping and Localization technology (SLAM) [6] [7] [8]. With the increase of computational power and advances in computer vision, work on integrating SLAM into indoor navigation has already been conducted [6] [9]

This concludes that to cater the needs of indoor navigation, a hypothetical system would require a more accurate user tracking than what GPS can offer, and with advances of modern technology, ideally look for opportunities to limit hardware units through using modern computer vision based solutions. In the case of retail or manufacturing contexts, the hypothetical navigation system should be able to define specific target destinations within accuracy of centimetres such that individual items or shelving units can be accurately localised. Commercial attempts to solve indoor navigation have also been made by commercial companies *Tangar* and *Mappedin*[10] [11]. Although information on technology used by *Tangar* and *Mappedin* is limited, their solutions fail to claim cross-platform compatibility with modern AR head-mounted displays (HMDs). Research suggests that in the context of navigation, a handheld device should be swapped for an HMD to reduce the task workload on end users [12], thus allowing them to freely use their hands while performing leisure or work related activities. This raises a research question of how can a scaleable, flexible AR navigation system be designed that could support tracking and guidance accuracy within centimetre range while supporting a HMD and also smartphones. While AR HMDs such as Microsoft HoloLens 1 and 2, Magic Leap 1 are expensive, bulky and largely inaccessible to an average consumer [13] [14], the market is moving towards more ergonomic and aesthetic AR solutions [14] [15] [16]. As of now, ideally an indoor navigation system should also be able to support smartphones due to their mainstream accessibility although a cross-platform compatibility would future-proof the software framework for when AR headsets also become a norm in our everyday life. The vision is that businesses can provide indoor navigation as a service, thus allowing customers and employees consume it using their device of choice.

1.1 Goal and Purpose

The goal is to develop a baseline novel navigation system that is not strictly platform dependant and with minimum effort can be adapted to a multitude of use-cases. This novel navigation system would solve the problem of high fidelity indoor navigation and provide navigation as a service by being agnostic of the end-user's device platform. Another goal is to measure the efficiency of this system through evaluating quantitative and qualitative data gathered from naïve test subjects using the system. This would provide a baseline solution for a more niche problem of indoor navigation

The purpose of the novel AR navigation system is to provide scalability and flexibility through semi-automated adaptation to an arbitrary environments and by providing a visual guidance of a shortest path between arbitrary targets given a set of destinations.

$$[x_{targets} \in Y_{destinations}]$$

2 Theoretical Background

2.1 Development of Augmented Reality

Augmented Reality enhances the real physical world with digital overlays such as graphics, sound or other sensory stimuli. It can be used in various areas, e.g. in games, in education, medical, navigation or industrial applications [13] [14] [17].

AR can be experienced over a multitude of platforms and devices. Devices running Android and iOS have the capability to host AR experiences through annotating the pass-through camera feed with holograms and make use of certain device's sensors. For Android and iOS there is support for AR software development kits (SDKs) such as ARCore and ARKit that directly integrate in modern game development engines such as Unreal or Unity [18] [19]. Similarly with state-of-the-art AR HMDs such as Magic Leap 1 and HoloLens 2 use Unity and Unreal are the main development engines that integrate each individual devices' SDK [18] [19]. To provide a seamless cross-platform compatibility, *Unity Technologies* provides a development framework called *AR Foundation* that acts as a layer of abstraction for each individual platform's SDK [20].

2.2 Environment Digitization

For a navigation system that could optimize routes and present a user with destinations, it is first important to create the map itself by digitizing the environment. The digitized environment (scene) can be described through a scene graph structure serializing into a JSON XML format. The scene describes the real-life environment's geometry with a finite resolution, as well as contains entities that describe different destinations, i.e. destination location, or their type. Representing an environment with such a structured approach allows other processing modules such as the deserializer and pathfinder to parse the data and process each entity according to its data type. An environment's meta model is shown in Figure 2.

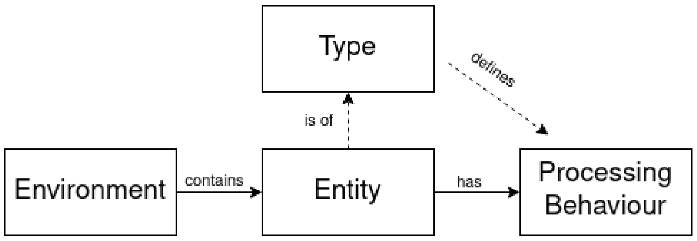


Figure 2: A digitized environment's meta model

2.2.1 Geometry Representation

Point clouds are essentially a collection of discrete points in space that collectively describe the shape of an arbitrary geometry [21]. Point clouds can be generated with the help of either lasers or RGB-D cameras. Industrial grade LiDAR scanners such as FARO Focus or Leica HDS6000 provide high levels of accuracy although they are expensive thus making them less accessible. A cheaper and more accessible solution is to use a an RGB-D camera, such as a Microsoft Kinect [22]. Other state of the art approaches use Simultaneous Localization and Mapping (SLAM) technology to generate a point cloud from a raw camera feed [7], a great benefit of this technology is that it can also be used to localize the camera in a 3D space.

Triangular meshes represent geometrical surfaces by sampling a point cloud and defining polygons between these faces [23]. As triangular meshes represent a surface, certain techniques exist for reducing the mesh data as this becomes mroe critical for decreasing the computational cost when processing the meshes [24].

2.2.2 Navigation Destination Representation

In a hypothetical navigation system, a destination is an arbitrary point in space with certain metadata that describes it. It can be a shelf or a particular item in a grocery store or a warehouse, a room number in an office space, or other. An automated approach for deriving destinations would be through applying a machine learning model that is trained to recognize different parts of a mapped environment and capture this metadata on a host device's memory.

When using the SLAM technology, also a framework exist for adding semantic labels to object during the environment mapping itself [25] [26], such approach would speedup the environment authoring time by shifting the object recognition away from a post-processing stage.

Fiducial marker based solutions can provide a reasonably simple but reliable tagging of certain areas within an environment [27] [28]. Common marker types are QR or ArUco that are used for tagging spaces. While this solution would be the most simplest, it would require manual efforts of placing the markers around areas of interest and associating their encoded information with certain labels.

2.3 User Tracking

The user tracking module provides an estimate of user's position in real time. Depending on the software design, it provides continuous user position updates such that a path finding module can keep optimizing the most optimum route in case the user's movement deviates from it. As GPS is not effective for indoors, the following sections give an overview of other available techniques that can be used individually or in tandem for better accuracy and reliability.

2.3.1 Beacons

A beacon enables broadcasting small pieces of information, i.e orientation data such as acceleration and rotation. In addition, the technology of the Bluetooth Low Energy (BLE) can be used, so that the device consumes little energy and a long battery life can be achieved. Therefore, the device transmits data in a low frequency range, specifically 2.4 GHz, which is also used by Bluetooth and WiFi [29]. One of the biggest differences between Bluetooth and BLE technology is that BLE is constantly in sleep mode, except when communicating, which also saves battery [30].

BLE beacons can be distinguished between connectable and non-connectable devices. The non-connectable beacons can only send data, whereas the connectable beacons can also receive and process data. Since only the transmission of tracking data is relevant for this project, the non-connectable beacons would be sufficient. By outputting the orientation data some aspects need to be taken into account, so that the correct data is also processed further. For example, attention must be paid to interference with the WiFi signal. Here, for instance, the method of fingerprinting could be used, where a previously selected signal is sent before the orientation data and an algorithm must be implemented on the receiver side, which recognizes the desired signal and processes the subsequent orientation data [29]. Since the BLE beacon technology was not used, this report does not go into further detail about the individual difficulties and implications.

2.3.2 Visual SLAM

Visual Simultaneous Localization and Mapping (VSLAM) is a vision-based tracking method [6]. It uses images acquired from a camera or other image sensors to perform location and mapping functions when the environment and location of the sensor are unknown. Originally intended for robot navigation, SLAM was later also used for AR, since no special algorithm or software is needed. Forms of VSLAM include monocular SLAM, binocular SLAM and RGB-D depth camera SLAM [31] [32].

Visual SLAM algorithms are broadly divided into two categories. These are depth methods, which use the brightness of the camera, and sparse methods, which match feature points of images [33]. Constructing a 3D map with VSLAM thus requires good lighting conditions and enough detectable visual features in indoor spaces. Bare walls or reflective surfaces severely limit accuracy [34]. To reduce accumulation errors and counteract fast camera movements, the extraction of points features can be extended by that of line features [34] [35]. However, as noted in other research, this method is not as well-established in literature and therefore poses a greater challenge [35].

2.3.3 Wifi-RTT

WiFi Round Trip Time, also known as WiFi RTT, operates similar to Beacons and allows devices such as smartphones to estimate the distance to a nearby WiFi access point (AP). The distance is determined directly on the device. WiFi RTT is based on the fine time measurement (FTM) framework. First, an FTM request is sent to the RTT AP. If this is accepted by the AP, the devices exchange the FTM-message. They store their transmit timestamps and the receive timestamp of the acknowledgement packet with a resolution of picoseconds to determine the time of arrival (ToA) and time of departure (ToD) [30] [36]. These are needed to calculate the Time of Flight (ToF) of the signal from the transmitter to the receiver [4]. In the example, this means from the AP to the smartphone.

WiFi RTT is characterized by an accuracy of one meter [4] [30]. Another advantage of this method is that the device does not have to be connected to the AP, which preserves the user's privacy [30].

2.3.4 Opti-Tracker

Position tracking with the Opti-Tracker system is done with the help of infrared cameras and markers. The infrared cameras are placed at certain distances from each other on the outer area of the tracking space. The setup of the system can be seen in Figure **??**. If a marker is brought into this area, it will be recognized by the infrared cameras and the position in space (relative to the cameras) can be calculated. With the help of this technique and the amount of cameras, the position of the marker can be determined accurately up to 1 mm [37]. With the help of the markers, non-accessible areas can be declared quickly and simply. In example, if markers are placed on the edge of non walk-able areas, these can later be defined as non-accessible places. This can be helpful when navigating.

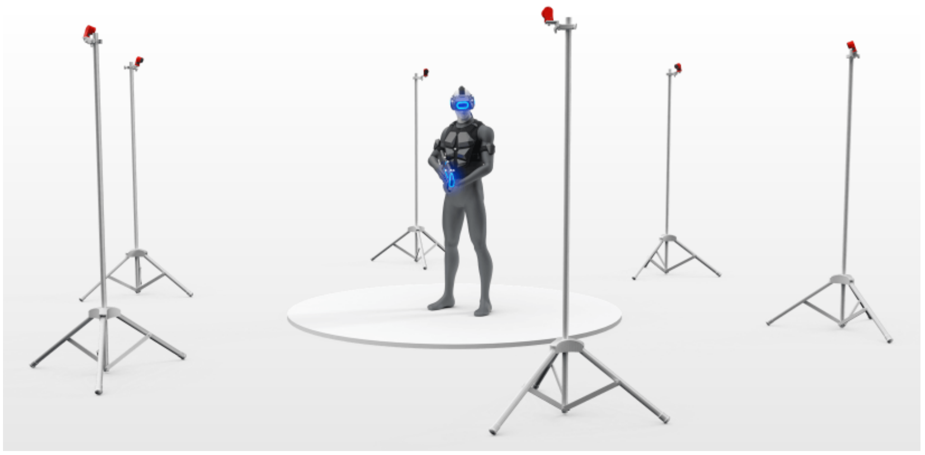


Figure 3: Setup of the Opti-Tracker system [37]

2.3.5 Kinect v2.0

The Kinect version 2.0 consists of an RGB- and an infrared-camera and three infrared emitters to guarantee the visibility of the environment. The infrared camera is responsible for depth estimation and both cameras record at 30 Hz. to obtain depth information, the Kinect v2.0 uses the time-of-flight or short T-o-F method. The infrared emitters send light in infrared wavelength, which is reflected by the objects in front of the camera and then hits the camera's sensors. From each of the 512×424 pixels, the distance between the object and the sensor is now measured, using the differences in time of flight, between the reflected signal and a reference signal received from the transmitter. With the help of the resulting phase shift, the distance can be calculated. The estimation of depth information, can be determined with a maximum field of view of 70° horizontal and 60° vertical [38]. According to Microsoft, precise measurements are ideal with a distance between sensor and object between 0.5 meters and 4.5 meters. Based on the depth information a point-cloud of the environment can be created to map out the space. To gain additional information about the position-tracking of people in the environment, the Kinect v2.0 can locate up to 6 people simultaneously and can output the position of the subjects in space, relative to its position [39].

2.3.6 Vive-Tracker

The HTC Vive-Tracker works similar to the Opti-Tracker. The tracking systems works with a base-station, also called Lighthouse, which emits infrared light. For this purpose, there is a tracker, which is equipped with photo-diodes on the outside. These are distributed at a constant distances from each other. In the following, the principle of this tracking method is briefly described and visualized in Figure 4.

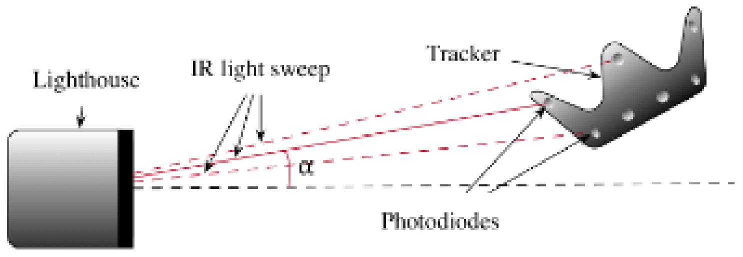


Figure 4: Operating principle of the Vive-Tracker [40]

The Lighthouse emits synchronized infrared light sweeps which triggers the photo-diodes of the tracker. Due to the delayed arrival time of the light sweeps at the photo diodes on the tracker, the horizontal and vertical angle can be determined. With several of these measurements, the position of the tracker in space can thus be determined. In addition, the tracker has a built-in Inertial Measurement Uni, which can be used to determine further movement and position data-updates to ensure a more accurate tracking [40].

2.4 Pathfinding

For the purpose of pathfinding, the A* algorithm is used. The algorithm aims to determine the shortest path from a starting point to the destination, via heuristic functions. The idea behind the algorithm is that it navigates a finite distance from node to node. Two other essential components are the two collections *open* and *closed* lists, they are used to sequentially connect the path. The flowchart in Figure 5 shows the operation of the A* algorithm [41].

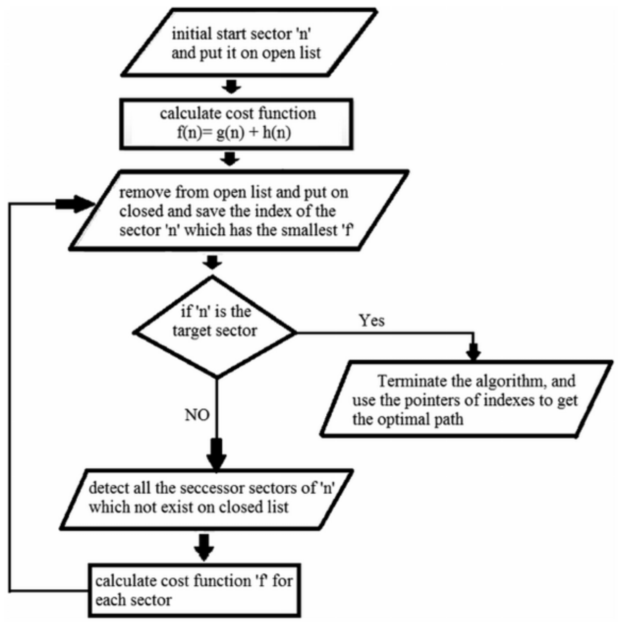

initial start sector 'n' and put it on open list
calculate cost function f(n)= g(n) + h(n)
remove from open list and put on closed and save the index of the sector 'n' which has the smallest 'f'
if 'n' is the target sector
Yes
Terminate the algorithm, and use the pointers of indexes to get the optimal path
NO
detect all the seccessor sectors of 'n' which not exist on closed list
calculate cost function 'f' for each sector

Figure 5: Flowchart of A* algorithm [41]

When the start and destination is set, the index of the start node is put in the open list and the destination is set as the target. Thereafter, every adjacent node is put in the open list and the cost function of each node gets calculated, with a heuristic equation which gets explained in the following part [42].

$$F = G + H \tag{1}$$

F = Total movement cost
G = Exact cost from the start node to current node n
H = Heuristic estimated cost from current node n to the end node

When calculating the G variable, it is possible to weight some movements. For example when a node is diagonal from the current, the G score can be weighted higher. This makes sense if moving diagonal should get punished. To estimate the H score there are different methods, but in this case the Manhattan method was used. So the Manhattan distance from the current node the the target, only moving horizontally and vertically, gets calculated and the amount of nodes that it takes to get there is summed up. The F score is the sum of the G and the H variable [43].

After calculating the F scores of the nodes in the open list, the node with the lowest F score gets chosen as the current node. The index of the previous current one is then put in the closed list. Thereafter it is checked if the current node is the target, if yes the algorithm is finished and the indices in the closed list, are the optimal path. If the current index does not refer to the target index, then all the adjacent nodes are put in the open list, except they are already in the closed list. Thereafter the process of calculating the F score and so on gets repeated until the target is reached.

3 Technical Background

3.1 Environment Processing

Given a certain environment description, a fully functional navigational system would require a processing engine that deserializes and interprets the data, performs route optimization and outputs guidance information over to client devices. This is envisioned as a centralized server and, in other works, is described as a *Navigational Module* [5].

Such a system would require wireless communication with client devices, it should be fast enough to calculate a shortest path in real-time and, in certain use cases, respond to dynamic changes in the environment such as misplaced retail items by customers.

3.2 Spatial Computing

3.3 AR Devices

Our framework's design will mainly focus around using state-of-the-art AR HMD's for displaying the visual guidance cues. This allows us to potentially improve comfort and usability of the system [12] and also to explore a different approach than what commercial products use [10] [11]. Currently in the market there are two options for standalone devices that provide a rich featureset, HoloLens 2 and Magic Leap 1. A feature overview can be found in the Table 1. Disregarding factors such as component or build quality, HoloLens 2 and Magic Leap 1 have a very similar baseline featureset. It is especially important to note that both devices support spatial mapping and six degree-of-freedom (6DoF) tracking. Meaning that a triangular mesh can be generated by scanning real-life geometry, and users can be tracked via their position and rotation.

	HoloLens 2	Magic Leap 1
Spatial Mapping	X	X
Eye tracking	X	X
Voice input	X	X
Semantic Labelling	X	X
Hand input	X	X
Custom hand gestures		X
Accelerometer/Gyroscope	X	X
WLAN	X	X
6DoF Tracking	X	X
Field of View (° diagonal)	54	50

Table 1: Brief overview of HoloLens 2 and Magic Leap 1 features [13] [14]

This means that both AR HMDs can be used as client devices but also as an environment authoring device.

Figure 6: Magic Leap 1, Source: [14]

3.3.1 Magicverse SDK

The Magicverse SDK builds on existing capabilities of the Magic Leap (ML) hardware and software, such as spatial mapping and cross-platform anchoring of content through Persistent Coordinate Frames (PCFs). It can be used to create multi-user applications for physical spaces, which are accessible via XR-compatible devices [44]

3.4 Unity

Unity is primarily a video game engine that for developing 2D and 3D video games but can also be used for immersive content, animation, architectural visualization or even machine learning and other use cases [45]. The editor is compatible with Mac and Windows as well as Linux and it supports building projects targeted for a multitude of platforms, including Lumin OS and Universal Windows Platform. The advantage of Unity is its ease of use, large community base and the ability to quickly iterate and prototype across development cycles through a rich proprietary API.

Magic Leap and HoloLens have also simulator frameworks available that integrate with Unity. This allows for triggering device specific events thus prototypes can be tested without having to build the app and deploy it to ML [44] [46]. Unity also provides a comprehensive collection of paid/free third-party assets through their asset store that increase developer productivity. Relevant assets for navigational systems would be networking (extOSC) and computer vision (OpenCV) [47].

4 Experimental Setup

Based on the evaluation of current state-of-the-art solutions and available hardware devices, we propose an experimental software setup that can be viewed as an initial prototype for the next generation indoor space navigation systems. The software architecture is based around principles of simplicity, cross-platform compatibility, flexibility, speed of environment authoring and scalability. The architecture is split into three separate components **Environment Mapping - Navigation Processor - Client Device**. An overview of a proposed architecture is depicted in Figure 7.

4.1 Environment Mapping

To generate paths, indoor navigation must be able to recognize the environment. Magic Leap's Spatial Mapper, which comes with the Lumin SDK, represents the geometry of the real environment as a single spatial map. It uses computer vision to recognize surfaces in real time and convert them into a large number of triangle meshes [14]. After an entire area is meshed, it can be spatially extracted.

Using computer vision, the Magic Leap can also recognize certain types of markers and infer their encoded IDs and estimate their position, and rotation in space. For our application, we used ArUco markers, which are binary, fiducial markers. To unify the coordinate space between mapping and client applications, a marker with a certain ID was used to which relative positions of all other objects were calculated to.

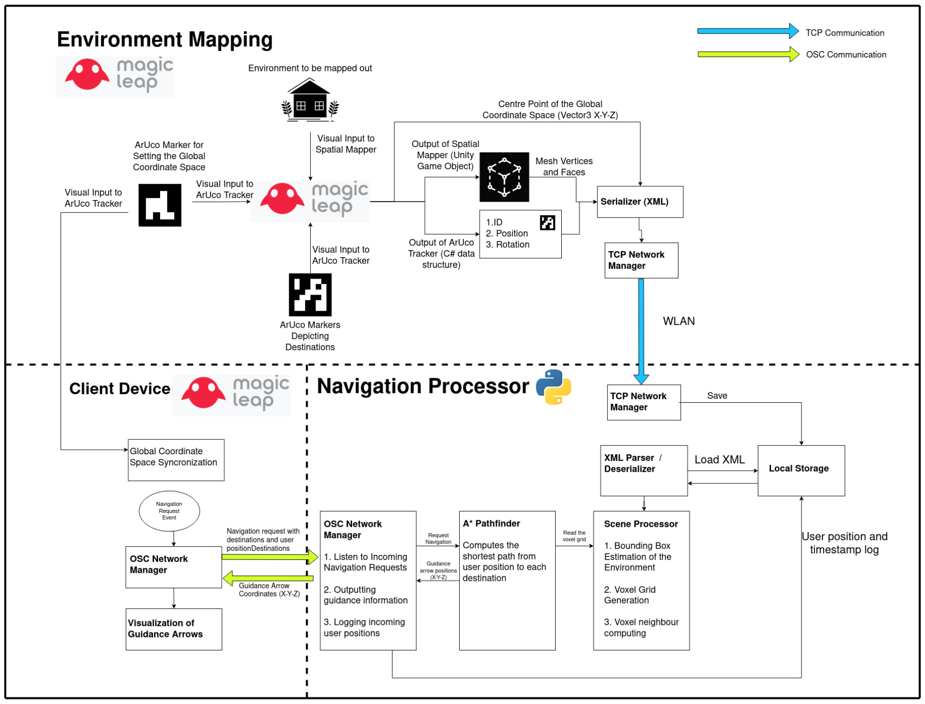

Environment Mapping
TCP Communication
OSC Communication
Environment to be mapped out
Centre Point of the Global Coordinate Space (Vector3 X-Y-Z)
Visual Input to Spatial Mapper
Output of Spatial Mapper (Unity Game Object)
Mesh Vertices and Faces
ArUco Marker for Setting the Global Coordinate Space
Visual Input to ArUco Tracker
Visual Input to ArUco Tracker
Serializer (XML)
1.ID
2. Position
3. Rotation
Output of ArUco Tracker (C# data structure)
TCP Network Manager
Visual Input to ArUco Tracker
ArUco Markers Depicting Destinations
WLAN
Client Device
Navigation Processor
TCP Network Manager
Save
Global Coordinate Space Syncronization
XML Parser / Deserializer
Load XML
Local Storage
Navigation Request Event
Navigation request with destinations and user positionDestinations
OSC Network Manager
Guidance Arrow Coordinates (X-Y-Z)
OSC Network Manager
1. Listen to Incoming Navigation Requests
2. Outputting guidance information
3. Logging incoming user positions
Request Navigation
Guidance arrow positions (X-Y-Z)
A* Pathfinder
Computes the shortest path from user position to each destination
Read the voxel grid
Scene Processor
1. Bounding Box Estimation of the Environment
2. Voxel Grid Generation
3. Voxel neighbour computing
User position and timestamp log
Visualization of Guidance Arrows

Figure 7: Overview of the proposed experimental setup's architecture

Unity's meshing snippet was used to add the real mesh objects as child objects to a parent object during runtime to aid the merging of all meshes into one upon serialization [44]. As soon as the environment has been scanned and all markers have been detected, an event is triggered via the user interface in the ML application in order to establish a connection to a server application via Wireless Local Area Network (WLAN). Subsequently, the environment data is sent over a Transmission Control Protocol (TCP) connection as an XML string. The string contains the environment in as a single node and also describes the orientation values of each scanned marker as individual nodes. Position and rotation values described in the scene description are relative to its parent node, see See listing 1.

Listing 1: Example Environment Description

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Root>
  <Mesh id="mesh:Meshes" pos="(0.0, 0.0, 0.0)" rot="(0.0, 0.0, 0.0)">
    <Vertex index="0" position="0.5 -0.5 0.5" />
    <Vertex index="1" position="-0.5 -0.5 0.5" />
    <Vertex index="2" position="0.5 0.5 0.5" />
    <Vertex index="3" position="-0.5 0.5 0.5" />
    <Vertex index="4" position="0.5 0.5 -0.5" />
    <Vertex index="5" position="-0.5 0.5 -0.5" />
    <Vertex index="6" position="0.5 -0.5 -0.5" />
    <Vertex index="7" position="-0.5 -0.5 -0.5" />
    <Face index="0" vertices="0 2 3" />
    <Face index="1" vertices="0 3 1" />
    <Face index="2" vertices="2 4 5" />
    <Face index="3" vertices="2 5 3" />
    <Face index="4" vertices="4 6 7" />
    <Face index="5" vertices="4 7 5" />
    <Face index="6" vertices="6 0 1" />
    <Face index="7" vertices="6 1 7" />
    <Face index="8" vertices="1 3 5" />
    <Face index="9" vertices="1 5 7" />
    <Face index="10" vertices="6 4 2" />
    <Face index="11" vertices="6 2 0" />
  <Mesh/>
  <Marker id="5" pos="(3.9, -0.1, 0.0)" rot="(0.0, 0.0, 0.0)" />
  <Marker id="15" pos="(2.6, -0.1, 0.0)" rot="(0.0, 0.0, 0.0)" />
  <Marker id="6" pos="(3.0, 0.1, -0.9)" rot="(0.0, 0.0, 0.0)" />
</Root>
```

4.2 Navigation Processor

The navigation processing module is responsible for handling the incoming environment data, saving it in local storage and deserializing the text into datas-

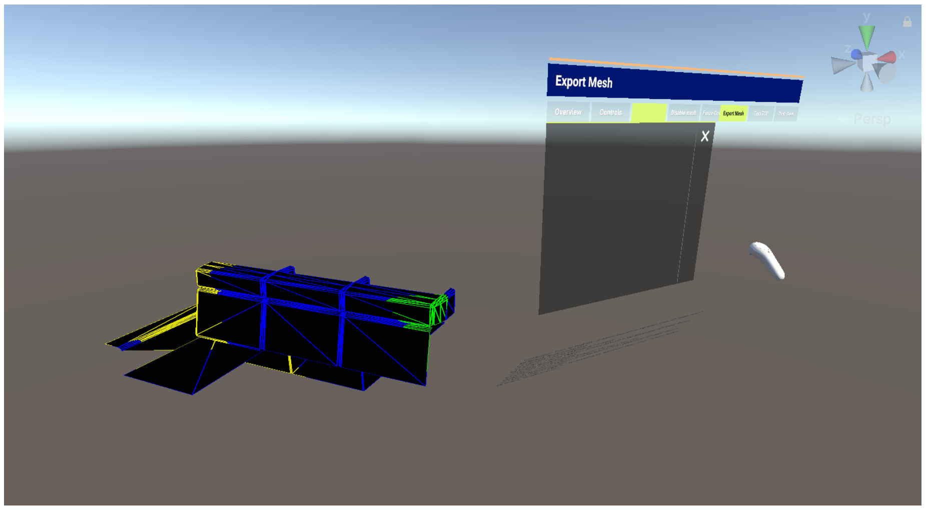


Figure 8: Spatial Mapping in Unity

tructures. The experimental module is written in Python 3.9 due to Python's flexibility, speed of prototyping and vast support for third party libraries.

The deserialization of XML was done using ElementTree library [48] and for creating datastructures that represent the scene, TriMesh library was used [49]. As TriMesh uses OpenGL right-handed coordinate system, the module also performs conversion from left-hand coordinate system (Unity) to right-handed (OpenGL). TriMesh also has a native voxelization engine that can be used to split a given mesh into discrete regions with a configurable size, as well as provides utility functions for querying the grid for voxel indices that contain arbitrary points in space. This is important for the project because it allows to query the grid for information such as, voxel index of the user location, voxel indices of destinations.

A voxelization of the environment is also critical for the A* pathfinding algorithm because it requires a grid-like structure as input for determining the shortest path. This also means that the size of each voxel affects the precission of the navigation and that is a trade-off for faster processing times. In our system, we estimate a bounding box of the mesh, which then is voxelized. This allows for a very simple determination of each voxels' neighbours, which is metadata required by the pathfinding algorithm. Given that the pathfinding submodule will be provided with neighbouring voxels, the output path will only be able to display guidance information laterally and longitudinally with a 45 degree resolution. In our system, we used a voxel size of 35cm, that gives a maximal destination offset error of 17,5cm.

Once an environment has gone through pre-processing, the module is ready to serve client applications by binding to an OSC port and listening for incoming navigation requests. A navigation request simply contains the number of destinations, along with the starting position. After feeding this information to the pathfinder module, the A* algorithm computes each route individually between destinations in the form of X-Y-Z positions of voxels, and afterwards concatenates all routes together to form a single path. The output to client device contains an array of X-Y-Z positions returned over OSC. [50].

4.3 Client Application

For the prototype, the client application is implemented on Magic Leap platform for the sake of hardware re-usability and its accessibility. Essentially this module could be developed on any other platform as long as it can read ArUco markers and bind to a UDP port over wireless network for receiving navigational information.

The module is intended to provide the user with the ability to raise the necessary events for issuing a navigation requests to a set of destinations. Intended to interpret the guidance information given by the *Navigation Processor* by displaying a guidance arrow at each of the provided cartesian coordinate points. Before the application can be successfully used, it is necessary to calibrate the global point of reference by gazing at a specific ArUco marker ID 49 that was designated specifically for this purpose. It is very important that the marker has not been moved ever since the environment description has been authored because otherwise this introduces an error when displaying the guidance arrows with the length of the offset vector.

Upon issuing a client request and receiving the guidance arrows, the client application starts to continuously update the Navigation Processor of its whereabouts such that any updates to the currently displayed route can be visualized.

Guidance arrows were color coded with default color being cyan, green denoting starting position and red denoting the destination. The color code was introduced to visualize a better indication of different parts of the route. See Figures 10 and 9 for a visualization.

5 Experiment

5.1 Participants

A total of 10 participants between the ages of 22 and 37 took part in the experiment. The average age was 26.3 years. None of the participants reported motor health problems, nor do they have vision problems or other health limitations

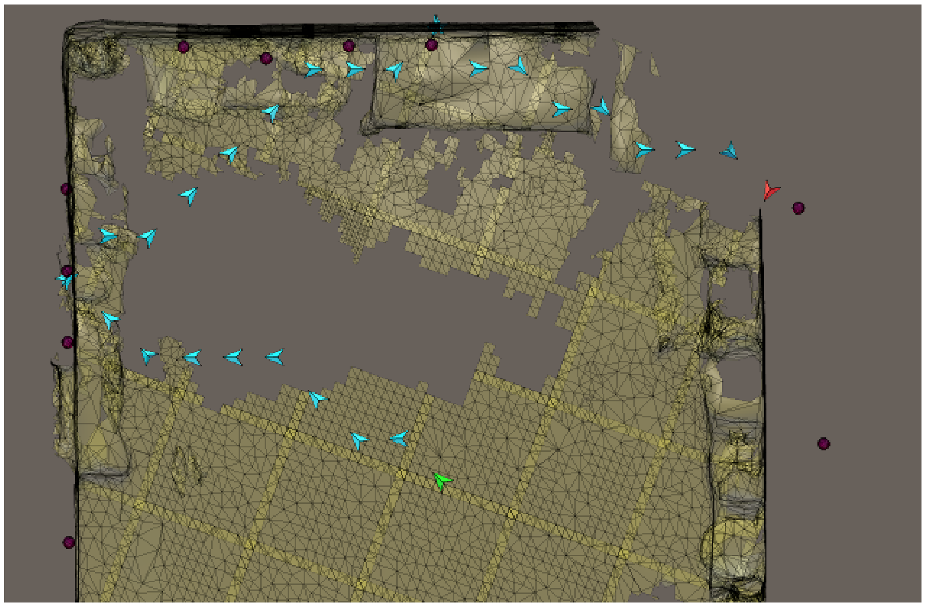


Figure 9: Mesh and Navigation Visualization. Isometric Top View.

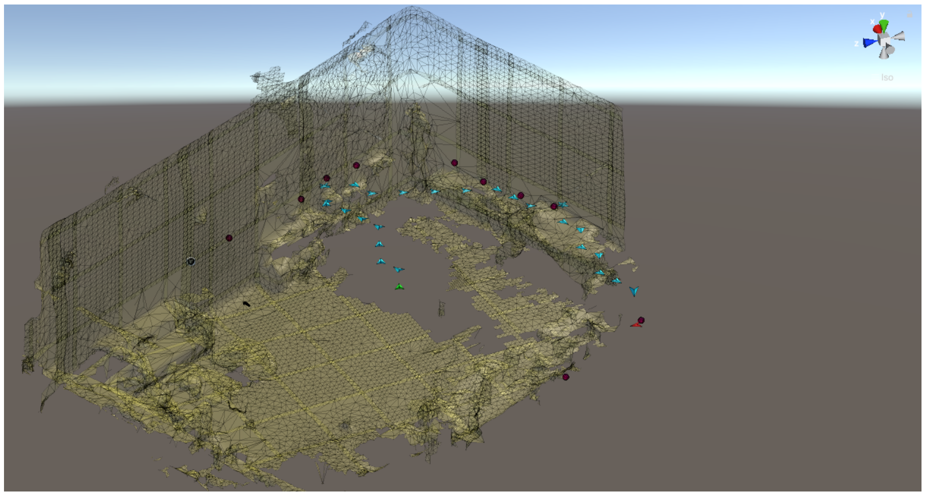


Figure 10: Mesh and Navigation Visualization. Isometric Perspective View.

that could affect the test. Half of the participants reported having no previous experience in AR.

5.2 Setup

11 ArUco markers with 11 different IDs were placed on the walls of a mock-up environment, see Figure 11. Some of the positional off-sets of markers is attributed to device's native inaccuracies when deriving their positions. Each test participant had a fixed starting point and was given the task of navigating to two pre-defined paths, once with and once without AR navigation. The path with AR navigation consisted of marker IDs 11, 0 and 7, and without AR navigation 5, 8 and 14. The identical paths for each test subject was intended to ensure a greater consistency among trials.

At each of the destinations, the test subjects were asked to mark it with a sticky-note before proceeding to the next destination. After the last marker was found, test subjects had to return to the starting point.

A total of two different assessments were conducted on the testing route. In the first assessment, participants used AR Indoor Navigation as an additional aid. In the second assessment, they had to complete the task without the navigation aid during which they were instructed that the order at which they visit each marker does not matter. During both test trials, each participants movement trajectory over time was recorded as a quantitative data measure. As qualitative data, after each test run the participants were asked to complete a questionnaire that is attached to this report.

6 Results

6.1 Questionnaires

In order to obtain valid results regarding the task workload, the questions of the NASA Task Load Index (NASA-TLX) were used. The questions provide information about the psychological stress, physical stress, time exposure, performance, exertion and frustration of the participants during each task. Workload related to the use of the system and contextual information is measured using a 5-point Likert scale. This is a psychometric response scale in which respondents indicate the extent to which they agree with a statement [51]. In our conducted test, the task workload should be assessed by the test takers as follows (1) Very Low; (2) Low; (3) Average; (4) High; (5) Very high. The arithmetic mean of each question is shown in Table 2. In addition, the Fleiss Kappa coefficient was calculated to determine the level of user agreement. This results in $k = 0.05$ for navigation with the Magic Leap and $k = 0.1$ for navigation without additional aids. In both cases, this corresponds to a slight agreement [52].

To investigate whether the results differed with and without the use of the Magic Leap, the **Wilcoxon Rank-Sum (WRS) test**, also known as Mann-

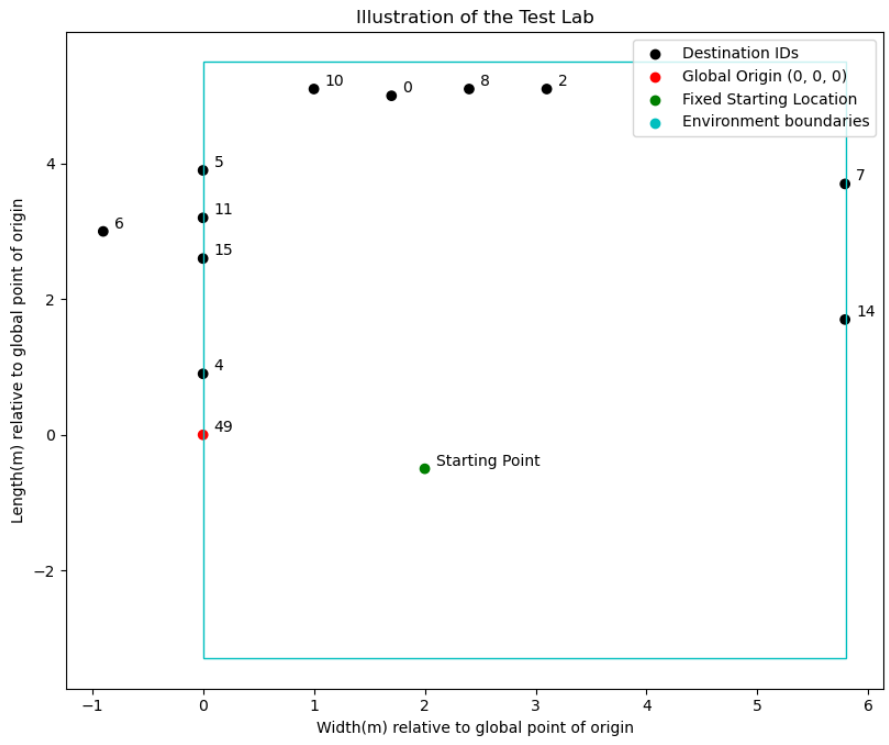


Figure 11: Illustration of the room marker setup from the data generated by the environment mapper

	Mean with ML	Std with ML	Mean without ML	Std without ML
How mentally demanding was the task?	1,8	0,8	2,0	0,6
How physically demanding was the task?	1,5	0,5	2,1	0,6
How hurried or rushed was the pace of the task?	2,2	0,7	2,7	0,8
How successful were you in accomplishing what you were asked to do?	2,6	1,2	3,7	0,8
How hard did you have to work to accomplish your level of performance?	2,4	0,9	2,3	0,7
How insecure, discouraged, irritated, stressed and annoyed were you?	2,6	0,7	1,7	0,7

Table 2: Overview of the NASA TLX questionnaire results

	With Navigation	Without Navigation
p0	0.446	0.238
Statistic	0.926	0.903

Table 3: Shapiro-Wilk Normality Test

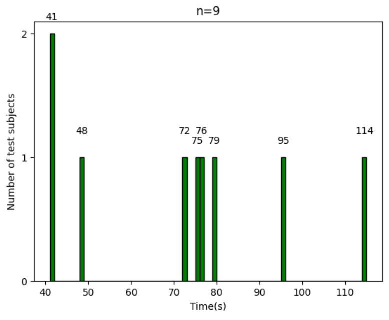
Whitney U test was used on the questionnaire results. The WRS test is often used in statistical practice for comparing measures when the distributions are not normal distributed or are not known [53]. One requirement is that the data be ordinally scaled, which is the case since a 5-point Likert scale was used. The results tell whether the central tendencies of two independent samples are different. So in this case, the question is whether using the Magic Leap makes a difference in terms of measuring the task workload.

For this, the respective (per question, with and without ML) supplemental rank sums R are formed. These are then compared with the critical value W. To determine the critical value, an alpha must be set, which in this case is put to 0.05. Then the number of answers given per question must be considered, which is 10. Afterwards, the critical value can be taken from a table, which in this case is 10. Is the critical value R smaller than the rank sum W it is assumed that no difference exists between the two measurement results. Is $W > R$ than it is assumed that there is a difference [54]. However, the test does not say why this difference between the measurement results occurs or does not occur [55].

The results show that in terms of mental demand R=19,5 and effort to accomplish the goal R=20,5, there is no difference whether the individuals used ML or not. But there are differences in physical demand R=6,5, temporal demand R=7, performance R=8 and feeling frustrated R=0.

6.2 Task Completion Time

As a quantitative data measure, the total task time was derived from each test participant's movement trajectory. A single data point was removed from the data set generated from the test with AR navigation, due to the very short task completion time of approximately two seconds. This can be explained by the test subject accidentally stopping the test without the test administrator noticing it. The raw data is depicted as histograms in Figures 12 and 13. The mean of total test time with AR navigation (n=9) is 71.3 seconds (std=23.44s), and the mean of total test time without AR navigation(n=10) is 46s (std=11.43s). Both data sets of with and without AR navigation were tested for normality using the Shapiro-Wilk test as it is the best fitting test recommended for small sample sizes n<50 [56]. Results of the normality test are depicted in 6.2



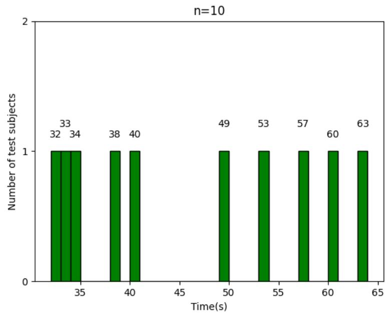
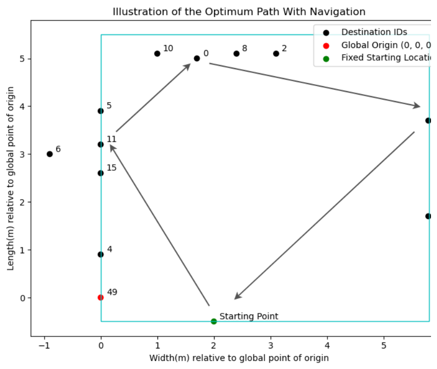


Figure 12: Test Completion Times With Navigation Aid

Figure 13: Test Completion Times Without Navigation Aid

6.3 User Position Tracking

Tracking data was also gathered of test participant position over the whole duration of the test. The raw data has been processed into heatmaps that give an insight into the movement trajectory of each test subject and how much time they've spent at a certain location. For a qualitative evaluation, Figures 16 to 21 depict comparisons of different test subject movement tendencies when conducting the test. Figures 14 and 15depict an ideal, best case scenario path.



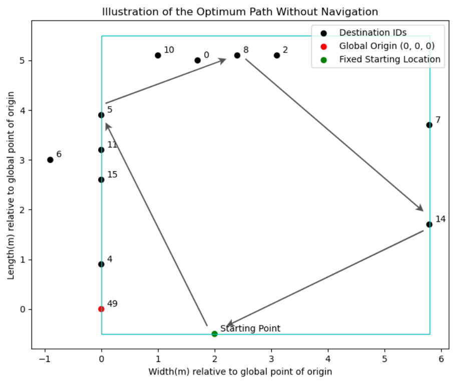


Figure 14: Most Optimum Path for Markers 11; 0; 7

Figure 15: Most Optimum Path for Markers 5; 8; 14

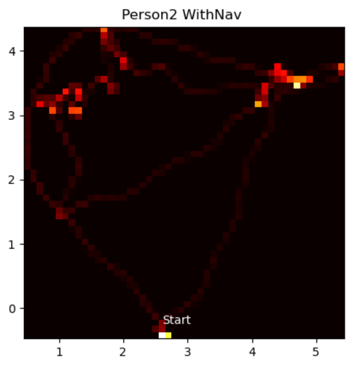


Figure 16: Participant Nr.2 Trajectory With Navigation

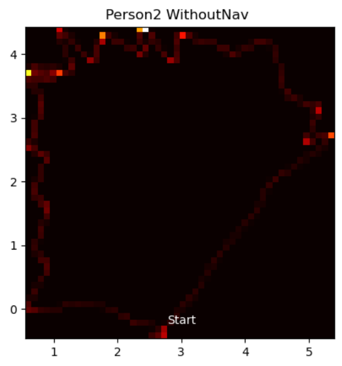


Figure 17: Participant Nr.2 Trajectory Without Navigation

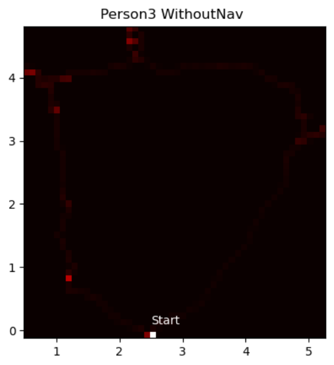


Figure 18: Participant Nr.3 Trajectory Without Navigation

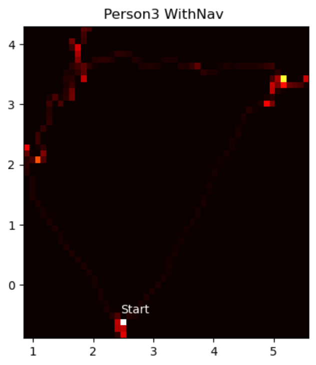


Figure 19: Participant Nr.3 Trajectory With Navigation

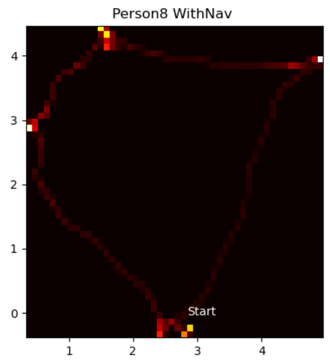


Figure 20: Participant Nr.8 Trajectory With Navigation

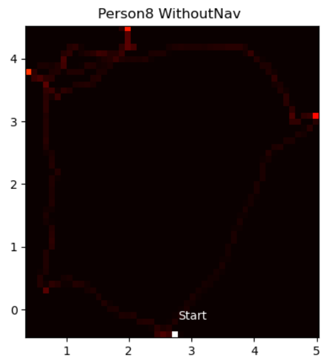


Figure 21: Participant Nr.8 Trajectory Without Navigation

7 Discussion

Given a 95% confidence interval, the Shapiro-Wilk normality test shows that both data sets of with and without AR navigation come from a normally distributed population $p0 > 0.05$. This indicates that our proposed solution for navigating between end destinations has worsened test subjects' ability to finish the test by 35%. This could be explained by a lack of understandable UI within the Client application as it was noted that many test subject appeared to get confused as to where exactly they need to go once the navigational information would appear. This suggests that our client application UX design should be rethought.

The calculation of the Fleiss Kappa coefficient showed that there was only a slight agreement between the test participants regarding the perception of the task workload between the two different assessments. This may be due to the fact that 50 % of the test participants already had experience in using AR devices, such as the ML, while the other 50 % had no touch points with such devices. Therefore, the subjective perceptions of test participants may differ.

The results show that navigation to different markers was barely mentally demanding for the test participants. Both scores are in the lower region of mental demand and the participants were a bit more united in the evaluation without ML, as shown by the smaller standard deviation. This could be explained by an overly simplified testing environment and straight forward task even. It was noted that during test without navigation, most of the test subjects simply went around the room in one direction, comparing the IDs with the destination list that they were given. This means that the test setup did not require the participants to develop their own strategies. Also, the results of the WRS test show that it made no statistical difference for participants to perform the task with or without ML (W=32.5>R=10).

The test participants found navigation with the application less physically demanding. Also the WRS test results show that there is a difference here (W=6.5<R=10). This is positive for the application. It is possible that this score is due to the fact that the participants were navigated directly to the targets and did not have to search all markers sequentially in order to reach their targets. So they didn't have to physically exert themselves. But it should also be said that in both cases the score is in the low range.

Also the WRS result supports that the participants felt a difference and were more rushed without the ML (W=7<R=10). This is an interesting result, also from the point of view that the participants performed the test with ML before the test without ML, so they already knew approximately how the test works and performed it once with ML. Nevertheless, they perceived the second run as more stressful. Of course, this may also be related to the fact that during the run with ML the targets were directly recognizable with the help of the arrows. Thus, they did not have to identify each marker individually.

When it comes to the aspect of how successful the participants felt in achieving the goals, a difference between the test runs becomes apparent. The WRS test results also confirms this difference (W=8<R=10). This shows that those without ML were more certain that they had achieved the goals. This could be related to the fact that 50% of the participants have no experience with AR devices and this affects their perception. This would also explain the higher std when using the ML, as the people with experience felt more confident to have reached the goals than the people who had not yet used an AR device.

When asked how hard the participants had to work to achieve their level of performance, there were no significant differences in the responses from the two test runs (W=20,5>R=10). This shows that in both cases an effort in the low range was enough to accomplish the level of performance. Similarly how with mental demand, this could also be attributed to the very simplistic test setup that did not properly challenge the test subjects.

The result of the last question shows that the participants felt more insecure, discouraged, irritated and stressed during the test run with the ML. Also the WSR test supports the difference (W=0<R=10). This result is surprising, since in previous results the physical demand and how stressed they felt were rated as lower with ML. The result may be related to the fact that, as already mentioned, half of the participants had no experience with AR devices and were therefore more irritated than in the test run without ML.

8 Conclusion and Future Work

Our solution proposes a first, working prototype of an indoor navigation system that proves to work starting from environment mapping down to pathfinding and guidance. Certain flaws were identified with the current design such as ArUco marker tracking inaccuracies, as poor Client Application UI, limited navigation guidance resolution and a very simplified test setup. Nevertheless this work provide basis for future work in the area by outlining a baseline architecture and providing insights of the first-iteration development cycle. Our solution allowed test subjects to consistently worsen their performance on finding the target destinations based on their higher total task time and higher standard deviation when using our developed AR navigational system. For the task load, certain questions had statistically significant differences between using the navigation and not, this tells that work should be continued on investigating the different navigation design in more challenging environments.

Our solution does not integrate any of the advanced features, such as misplaced item recognition by client devices through computer vision, IoT communication and rerouting to misplaced items by the Navigational Module or testing in a production environment with multiple devices. For further work on this project, research could be done on which guidance information is the most effective in guiding the user to their destinations the fastest. Another point would be to bring in computer vision to recognize targets and annotate. This way the user would immediately recognize the target and this could have a positive effect on the task completion time. Also this work does not investigate how auditory cues could be added to aid the overall task completion. In addition, on-boarding in terms of AR device usage would be possible, so that people who have never had contact with such devices become used to it and are not caught off guard or confused.

References

[1] Frank Van Diggelen and Per Enge. The World's First GPS MOOC and Worldwide Laboratory Using Smartphones. In *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)*, pages 361–369, 2015.

[2] Google Maps. `https://support.google.com/maps/`. Accessed: 2021-12-20.

[3] Apple Indoor Maps Program. `https://register.apple.com/resources/indoor/Apple-Indoor-Maps-Program.pdf`. Accessed: 2021-12-17.

[4] H Cao, Y Wang, J. Bi, S. Xu, M. Si, and H Qi. Indoor Positioning Method Using WiFi RTT Based on LOS Identification and Range Calibration. *International Journal of Geo Information*, 9(11):627, 2020.

[5] Jayakanth Kunhoth, AbdelGhani Karkar, Somaya Al-ma'adeed, and Abdulla Al-Ali. Indoor Positioning and Wayfinding Systems: A Survey. *Human-centric Computing and Information Sciences*, 10, 12 2020.

[6] H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping: Part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[7] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[8] Shibo Han, Minhaz Uddin Ahmed, and Phill Kyu Rhee. Monocular SLAM and Obstacle Removal for Indoor Navigation. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 67–76, 2018.

[9] Akshat Bajpai and Sepehr Amir-Mohammadian. Towards An Indoor Navigation System Using Monocular Visual SLAM. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 520–525, 2021.

[10] Indoor Navigation Using Computer Vision and AR. `https://tangar.io/`. Accessed: 2021-12-18.

[11] Indoor Maps and Wayfinding — Mappedin. `https://www.mappedin.com/`. Accessed: 2021-12-18.

[12] Renan Guarese and Anderson Maciel. *Development and Usability Analysis of a Mixed Reality GPS Navigation Application for the Microsoft HoloLens*, pages 431–437. 06 2019.

[13] Microsoft. `https://microsoft.com`. Accessed: 2021-12-20.

[14] Magic Leap. `https://www.magicleap.com/`. Accessed: 2021-12-20.

[15] Introducing Ray-Ban Stories: First Generation Smart Glasses — Meta. `https://about.fb.com/news/2021/09/introducing-ray-ban-stories-smart-glasses/`. Accessed: 2021-12-18.

[16] Nreal Light - Ready-To-Wear Mixed Reality Glasses. `https://www.nreal.ai/light`. Accessed: 2021-12-18.

[17] Jad Chalhoub and Steven K. Ayer. Using Mixed Reality for electrical construction design communication. *Automation in Construction*, 86:1–10, 2018.

[18] Unity - Manual: Getting Started With AR Development in Unity. `https://docs.unity3d.com/Manual/AROverview.html`. Accessed: 2021-12-18.

[19] AR Platforms — Unreal Engine Documentation. `https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/AR/ARPlatforms/`. Accessed: 2021-12-18.

[20] Unity Technologies. About AR Foundation — AR Foundation — 4.2.1, 2020.

[21] Marc Levoy and Turner Whitted. *The Use of Points as a Display Primitive*. Department of Computer Science, University of North Carolina, 1985.

[22] Felix Endres, Jurgen Hess, Nikolas Engelhard, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. An Evaluation of the RGB-D SLAM System. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1691–1696, 05 2012.

[23] Maarten Bassier, Maarten Vergauwen, and Florent Poux. Point Cloud Vs. Mesh Features for Building Interior Classification. *Remote Sensing*, 12(14):2224, 2020.

[24] Jerry O Talton. A Short Survey of Mesh Simplification Algorithms. *University of Illinois at Urbana-Champaign*, 2004.

[25] Xuxiang Qi, Shaowu Yang, and Yuejin Yan. Deep Learning Based Semantic Labelling of 3D Point Cloud in Visual SLAM. In *IOP Conference Series: Materials Science and Engineering*, volume 428, page 012023, 10 2018.

[26] Sudeep Pillai and John Leonard. Monocular SLAM Supported Object Recognition. 07 2015.

[27] Nida Romli, Amir Razali, Nur Hafizah Ghazali, Nik Adilah Hanin Binti Zahri, and Siti Ibrahim. Mobile Augmented Reality (AR) Marker-based for Indoor Library Navigation. *IOP Conference Series: Materials Science and Engineering*, 767:012062, 03 2020.

[28] Jack Cheng, Keyu Chen, and Weiwei Chen. Comparison of Marker-Based AR and Markerless AR: A Case Study on Indoor Decoration System. 07 2017.

[29] Joakim Lindh. Bluetooth Low Energy Beacons. *Texas Instruments*, page 2, 2015.

[30] Maroš Šeleng. WiFi Round Trip Time for Indoor Navigation. Master's thesis, Masaryk University, 2019.

[31] Kaichang Di, Wenhui Wan, H. Zhao, Zhaoqin Liu, R. Wang, and F. Zhang. Progress and Applications of Visual SLAM. *Cehui Xuebao/Acta Geodaetica et Cartographica Sinica*, 47:770–779, 06 2018.

[32] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9, 12 2017.

[33] What Is SLAM? 3 Things You Need to Know. `https://se.mathworks.com/discovery/slam.html`. Accessed: 2021-12-20.

[34] Huang J. Ma W, Zhang S. Mobile Augmented Reality Based Indoor Map for Improving Geo-Visualization. *PeerJ Computer Science*, 7:704, 2021.

[35] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. PL-SLAM: Real-time Monocular Visual SLAM with Points and Lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508, 2017.

[36] Christian Gentner, Markus Ulmschneider, Isabel Kuehner, and Armin Dammann. WiFi-RTT Indoor Positioning. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1029–1035, 2020.

[37] Optitrack. `https://optitrack.com/`. Accessed: 2021-12-11.

[38] P. Frankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart. Kinect v2 for Mobile Robot Navigation: Evaluation and Modeling. 2015.

[39] E. Lachat, H. Macher, M.-A. Mittet, T. Landes, and P. Grussenmeyer. FIRST EXPERIENCES WITH KINECT V2 SENSOR FOR CLOSE RANGE 3D MODELLING, 2015.

[40] Miguel Borges, Andrew Symington, Brian Coltin, Trey Smith, and Rodrigo Ventura. HTC Vive: Analysis and accuracy improvement.

[41] Issa Zidane and Khalil Ibrahim. *Wavefront and A-Star Algorithms for Mobile Robot Path Planning*, page 74. 10 2018.

[42] Xiao Cui and Hao Shi. A*-based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security*, 11, 2011.

[43] Patrick Lester. A* Pathfinding for Beginners. `http://www.policyalmanac.org/games/aStarTutorial.htm`. Accessed: 2021-12-20.

[44] Magic Leap Developer. `https://developer.magicleap.com/`. Accessed: 2021-12-20.

[45] Afzal Hussain, Haad Shakeel, Faizan Hussain, Nasir Uddin, and Turab Ghouri. Unity Game Development Engine: A Technical Survey. *University of Sindh Journal of Information and Communication Technology*, 4, 10 2020.

[46] MRTK Simulator. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/input-simulation/input-simulation-service?view=mrtkunity-2021-05. Accessed: 2021-12-19.

[47] Unity Asset Store. https://assetstore.unity.com/. Accessed: 2021-12-17.

[48] The ElementTree XML API. https://docs.python.org/3/library/xml.etree.elementtree.html. Accessed: 2021-12-18.

[49] Basic Installation - TriMesh. https://trimsh.org/. Accessed: 2021-12-18.

[50] A Guide to Unity's Coordinate System (With Practical Examples). https://www.techarthub.com/a-guide-to-unitys-coordinate-system-with-practical-examples/. Accessed: 2021-12-18.

[51] Victor R. Preedy and Ronald R. Watson, editors. *5-Point Likert Scale*, chapter 10, pages 4288–4288. Springer New York, New York, NY, 2010.

[52] Philipp Schaer, Philipp Mayr, and Peter Mutschke. Implications of Inter-Rater Agreement on a Student Information Retrieval Evaluation. *arXiv preprint arXiv:1010.1824*, 2010.

[53] J. Perolat, I. Couso, K. Loquin, and O. Strauss. Generalizing the Wilcoxon Rank-Sum Test for Interval Data. *International Journal of Approximate Reasoning*, 56:108–110, 08 2015.

[54] Wilcoxon Rank Sum Test for Independent Samples. https://www.real-statistics.com/non-parametric-tests/wilcoxon-rank-sum-test/. Accessed: 2021-12-17.

[55] Mann-Whitney-U-Test. https://www.methodenberatung.uzh.ch/de/datenanalyse_spss/unterschiede/zentral/mann.html. Accessed: 2021-12-17.

[56] Asghar Ghasemi and Saleh Zahediasl. Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International journal of endocrinology and metabolism*, 10:486–489, 12 2012.